LISTING OF CLAIMS

1. (currently amended) A memory pattern for a memory device for use by a computer system, wherein the memory pattern is obtained after a function is called when said computer system executes a program, said memory pattern comprising:

a return address storage area for storing a return address for the source of a call for the execution of a currently active function;

a previous frame pointer storage area for storing a previous frame pointer to said calling source for the execution of a currently active function; and

a local variable storage area to be located below said return address storage area and said previous frame pointer storage area,

wherein when a data array is stored in said local variable area, a guard variable is stored in a location preceding said data array between said frame pointer and said data array, and

wherein said guard variable is used as a target to confirm whether said return address has been destroyed.

2. (original) The memory device according to claim 1, wherein, when a character string is stored in said local variable storage area in said memory pattern of said memory device, said guard variable is stored preceding said character string.

3. (original) The memory device according to claim 1, wherein a random number can be employed as said guard variable that is stored in said local variable storage area in said memory pattern of said memory device.

4. (currently amended) A memory device for use by a computer system, said memory device having a memory pattern obtained after a function is called while said computer system is executing a program, said memory pattern comprising:

a return address storage area for storing a return address for the source of a call for a function that is currently being executed;

a previous frame pointer storage area ~~is~~ located below said return address storage area for the storage of a previous frame pointer for the source of said call; and

a local variable storage area ~~is~~ located below said previous frame pointer storage area,

wherein when a data array is stored in said local variable storage area, a guard variable is stored between said previous frame pointer and said data array, and

wherein said guard variable is used as a target to confirm whether said return address has been destroyed during a process for returning from said function that is currently being executed.

5. (original) A stack protection system, which protects a program counter from a stack smashing attack while a computer is executing a program, comprising:

a stack protection instruction preparation unit for receiving a source program, and for adding, to said source program, an instruction for storing a guard variable between a previous frame pointer in a stack that is obtained after a function has been called, and a data array stored in said stack as a local variable; and

a stack protection execution unit for executing said program to which said instruction for storing said guard variable is added by said stack protection instruction preparation unit, and for, in accordance with said instruction, storing said guard variable in said stack during a function calling process and for confirming the validity of said guard variable during a function return process.

6. (original) The stack protection system according to claim 5, wherein, when destruction of said guard variable is discovered during said function return process, said stack protection execution unit performs an abnormal end process to halt the execution of said program, and to notify a user of an occurrence of a stack smashing attack.

7. (original) The stack protection system according to claim 5, wherein said stack protection instruction preparation unit is mounted in a compiler that processes said source program written in a compatible language, when said source program is translated into an object program, the compiler adds to said object program an instruction for the storage of said guard variable.

8. (original) A stack protection system, which protects a program counter from a stack smashing attack when a computer executes a program, comprising:

program execution means, for receiving and executing a program to which an instruction is added for the storage, before a function is executed, of a guard variable between a previous frame pointer in a stack and a data array stored as a local variable in said stack, and for, in accordance with said instruction, storing said guard variable in said stack during a function calling process, and for confirming the validity of said guard variable during a function return process; and

abnormal end execution means for, when said program execution means detects during said function return process that said guard variable has been destroyed, performing an abnormal end process for halting the execution of a program and for notifying a user that a stack smashing attack has occurred.

9. (currently amended) A computer system comprising:

a data processor for performing various calculations; and

a memory device that is used by said data processor for said calculations and that has a memory pattern that is obtained after a function is called while said computer system is executing a program, said memory pattern including:

a return address storage area for storing a return address for the source of a call for a function that is currently being executed,

a previous frame pointer storage area for storing a previous frame pointer for said source of said call for the function that is currently being executed, and

a local variable storage area that is located below said return address storage area and said previous frame pointer storage area,

wherein, when a data array is stored in said local variable storage area, a guard variable is stored preceding said data array in said local variable area, between said frame pointer and the data array, and

wherein, said guard variable is used as a target to confirm, during a process for the return of said function that is currently being executed, whether said return address has been destroyed.

10. (original) A computer system comprising:

a data processor for performing various calculations; and

a memory device used by said data processor for said calculations, that has a memory pattern that is obtained after a function is called while said computer system is executing a program, said memory pattern including

a return address storage area for storing a return address for a source that called a function that is currently being executed,

a previous frame pointer storage area that is located below said return address storage area for the

storage of a previous frame pointer for said source that called said function, and

a local variable storage area that is located below said previous frame pointer storage area,

wherein when a data array is stored in said local variable storage area, a guard variable is stored between said previous frame pointer and said data array, and

wherein said guard variable is used as a target to confirm whether said return address has been destroyed during a process for returning from said function that is currently being executed.

11. (original) A computer system, which controls a program for performing various calculations, comprising:

a data processor for reading and executing a program to which an instruction is added for the storage, when a function is to be executed, of a guard variable between a previous frame pointer in a stack and a data array stored as a local variable in said stack; and

a memory device used by said data processor to perform calculations,

wherein, in accordance with said instruction for storing said guard variable that is added to said program, said data processor stores said guard variable in said stack of said memory device during a function calling process, and confirms the validity of said guard variable during a function return process.

12. (original) The computer system according to claim 11, wherein, when destruction of said guard variable is

found during said function return process, said data processor performs an abnormal end process to halt the execution of said program, and to notify a user of the occurrence of a stack smashing attack.

13. (original) A computer system, which controls a program for performing various calculations, comprising:

a data processor for receiving a source program and for adding, to said source program, an instruction for storing a guard variable between a previous frame pointer in a stack that is obtained after a function is called, and a data array stored in said stack as a local variable, and for executing said program to which said instruction for storing said guard variable has been added; and

a memory device used by said data processor to perform said calculations,

wherein in accordance with said instruction for storing said guard variable said data processor stores said guard variable in said stack of said memory device during a function calling process, and confirms the validity of said guard variable during a function return process.

14. (original) The computer system according to claim 13, wherein, when destruction of said guard variable is discovered during said function return process, said data processor performs an abnormal end process to halt the execution of said program, and to notify a user of the occurrence of a stack smashing attack.

15. (original) A compiler, which receives a source program, translates said source program to provide an object program and then outputs said object program, comprising:

translation means for translating a program; and

stack protection instruction addition means for examining each sub-routine to determine whether a function included in said sub-routine has a data array, and for, when a data array is included, adding to said sub-routine an instruction for storing, before a function is executed, a guard variable between a previous frame pointer in a stack and a data array stored as a local variable in said stack, and for confirming the validity of said guard variable during a function return process.

16. (original) A stack protection method, whereby a program counter is protected from a stack smashing attack while a computer is executing a program, comprising the steps of:

adding, to a source program, an instruction for storing a guard variable between a previous frame pointer in a stack that is obtained after a function has been called, and a data array stored in said stack as a local variable;

executing said program to which said instruction for storing said guard variable is added, and, in accordance with said instruction, storing said guard variable in said stack during a function calling process and confirming the validity of said guard variable during a function return process;

when destruction of said guard variable is discovered during said function return process, halting the execution of said program and notifying a user of an occurrence of a stack smashing attack.

17. (original) A stack protection method, whereby a program counter is protected from a stack smashing attack when a computer executes a program, comprising the steps of:

executing a program to which an instruction is added for the storage, before a function is executed, of a guard variable between a previous frame pointer in a stack and a data array stored as a local variable in said stack, and, in accordance with said instruction, storing said guard variable in said stack during a function calling process and confirming the validity of said guard variable during a function return process; and

when it is detected during said function return process that said guard variable has been destroyed, halting the execution of a program and notifying a user that a stack smashing attack has occurred.

18. (original) A storage medium on which input means for a computer stores a computer-readable program, said program permitting said computer to perform:

a process for storing, before a function having a data array is executed, a guard variable between a previous frame pointer in a stack and a data array stored as a local variable in said stack;

a process for confirming the validity of said guard variable during a function return process; and

a process for, when it is detected during said function return process that said guard variable has been destroyed, halting the execution of a program and notifying a user that a stack smashing attack has occurred.

19. (original) A program transmission apparatus comprising:

storage means for storing a program that permits a computer to perform

a process for storing, before a function having a data array is executed, a guard variable between a previous frame pointer in a stack and a data array stored as a local variable in said stack,

a process for confirming the validity of said guard variable during a function return process, and

a process for, when it is detected during said function return process that said guard variable has been destroyed, halting the execution of a program and notifying a user that a stack smashing attack has occurred; and

transmission means for reading said program from said storage means and for transmitting said program.